A Principled Approach to Reasoning about the Specificity of Rules

John Yen
Department of Computer Science
Texas A&M University
College Station, TX 77843
Yen@CSSUN.TAMU.EDU

Abstract

Even though specificity has been one of the most useful conflict resolution strategies for selecting productions, most existing rule-based systems use heuristic approximation such as the number of clauses to measure a rule's specificity. This paper describes an approach for computing a principled specificity relation between rules whose conditions are constructed using predicates defined in a terminological knowledge base. Based on a formal definition about pattern subsumption relation, we first show that a subsumption test between two conjunctive patterns can be viewed as a search problem. Then we describe an implemented pattern classification algorithm that improves the efficiency of the search process by deducing implicit conditions logically implied by a pattern and by reducing the search space using subsumption relationships between predicates. Our approach enhances the maintainability of rule-based systems and the reusability of definitional knowledge.

Introduction

Specificity is a classic conflict resolution heuristic used by many rule languages from OPS through ART for selecting productions [McDermott and Forgy 1978]. It provides a convenient way for expert systems (such as R1) to describe general problem solving strategies as well as strategies for handling exceptional cases. In a similar spirit, common sense reasoning also relies on the specificity of a rule's antecedents to override conclusions drawn by more general rules when they contradict the more specific rule.

Even though the specificity of rules has been long recognized as an important information for the selection of rules, few efforts have been made to develop algorithms for computing a principled measure of rules' specificity. Instead, most existing rule systems use syntactic information such as the number of clauses as a heuristic approximation to the specificity of rules. This has both encouraged, and to some extent necessitated, bad programming practices in which clauses are placed in production rules solely to outsmart the conflict resolution algorithm. As a result, it is hard to explain rules and difficult to determine how to correctly add or revise them. Two other problems with rule-based systems have often been identified by critics. First, the meaning of the terminology used by rules is often ill-defined [Swartout and Neches 1986]. This makes it difficult to determine when rules are, or should be, relevant to some shared abstraction – which, in turn, makes it difficult to find and change abstractions. Third, it is difficult to structure large rule sets [Fikes and Kehler 1985]. This makes it difficult to decompose the set into smaller, more comprehensible and maintainable subsets.

To address these problems with rule-based systems, we have developed a production system, CLASP, where the semantics of predicates used in rules are defined using a term subsumption language (LOOM)[1] [Yen *et al.* 1989]. One of the major feature of CLASP is a *pattern classifier* that organizes patterns into a lattice where more specific patterns are below more general ones, based on the definitions of predicates in the patterns. Using the pattern classifier, CLASP can compute a well-defined specificity relation between rules.

Related Work

The idea of using the taxonomic structure of a terminological knowledge base to infer specificity relations between rules was first introduced by CONSUL[Mark 1981]. Because rules in CONSUL mapped one description to another, the condition of a CONSUL's rule is just a concept. Specificity of

[1] Term subsumption languages refer to knowledge representation formalisms that employ a formal language, with a formal semantics, for the definition of terms (more commonly referred to as concept or classes), and that deduce whether one term subsumes (is more general that) another using a classifier [Patel-Schneider *et al.* 1990]. These formalisms generally descend from the ideas presented in KL-ONE [Brachman and Schmolze 1985]. LOOM is a term subsumption-based knowledge representation system developed at USC/ISI [Gregor and Bates 1987].

rules can thus be obtained directly from the concept subsumption lattice. To verify the consistency and completeness of expert systems, researchers have also developed algorithms for detecting subsumed rules based on a subset test of clauses [Suwa *et al.* 1982, Nguyen *et al.* 1985]. More recently, the problem of computing the subsumption relation between plan classes has also been explored [Wellman 1988].

Defining Pattern Subsumption Relations

Conceptually, a pattern $P2$ is more specific than (i.e., is subsumed by) a pattern $P1$ if, for all states of the facts database, a match with $P2$ implies a match with $P1$. To define the subsumption of patterns more formally, we need to introduce the following terminology. A pattern is denoted by P_{X} where X denotes the set of variables in the pattern[2]. An instantiation of the pattern is denoted as $P_{\mathrm{X}}(\vec{x})$ where $\vec{x}$ is a vector of variable bindings for X. For instance, the expression $P1_{[?x1\ ?x2\ ?x3]}([John\ Angela\ Car1])$ denotes an instantiation of P1 that binds pattern variables ?x1, ?x2, and ?x3 to John, Angela, and Car1 respectively. Let $\mathcal{T}$ be a terminological knowledge base. Concepts and roles (i.e., relations) are unary predicates and binary predicates defined in $\mathcal{T}$. An interpretation $\mathcal{I}_{\mathcal{T}}$ of $\mathcal{T}$ is a pair $(\mathcal{D}, \mathcal{E})$ where $\mathcal{D}$ is a set of individuals described by terms in $\mathcal{T}$ and $\mathcal{E}$ is an *extension function* that maps concepts in $\mathcal{T}$ to subsets of $\mathcal{D}$ and roles in $\mathcal{T}$ to subsets of the Cartesian product, $\mathcal{D} \times \mathcal{D}$, denoted as $\mathcal{D}^2$. $P^{\mathcal{E}}(\vec{x})$ denotes that $\vec{x}$ satisfies the condition of the pattern P **under the extension function** $\mathcal{E}$, i.e., $\forall x, y \in \mathcal{D}$

- $(C\ x)^{\mathcal{E}}$ iff $x \in \mathcal{E}(C)$
- $(R\ x\ y)^{\mathcal{E}}$ iff $[x, y] \in \mathcal{E}(R)$
- $(l_1 \wedge l_2)^{\mathcal{E}}$ iff $l_1^{\mathcal{E}} \wedge l_2^{\mathcal{E}}$

where C and R denote concepts and relations defined in $\mathcal{T}$; l_1 and l_2 denote two literals.

Definition 1 *Suppose* $P1_{\mathrm{Y}}$ *and* $P2_{\mathrm{X}}$ *are two patterns whose predicates are defined in a terminological knowledge base* $\mathcal{T}$. *The pattern* $P1_{\mathrm{Y}}$ *subsumes* $P2_{\mathrm{X}}$, *denoted as* $P1_{\mathrm{Y}} \succeq P2_{\mathrm{X}}$, *iff*

$$\forall\, \mathcal{I}_{\mathcal{T}} = (\mathcal{D},\ \mathcal{E})\ \forall \vec{x} \in \mathcal{D}^n$$
$$\left(\ P2_{\mathrm{X}}{}^{\mathcal{E}}\ (\vec{x}) \Rightarrow \exists\ \vec{y} \in \mathcal{D}^m\ P1_{\mathrm{Y}}{}^{\mathcal{E}}\ (\vec{y})\ \right) \qquad (1)$$

where $\vec{x}$ *and* $\vec{y}$ *are vectors of elements in* $\mathcal{D}$, *with dimension n and m respectively.*

It is easy to verify that the pattern subsumption relation is reflexive and transitive.

Several important points about our definition of the pattern subsumption relation are worth mentioning. First, the definition allows patterns with different number of variables to be compared with each other. This is important for using the subsumption of patterns as a useful measure of the specificity of rules, for the condition of a specific rule often introduces extra variables to test a situation that is more complicated than the condition of a general rule. Enforcing that two subsuming patterns have same number of variables will limit the usefulness of pattern subsumption taxonomy for controlling the firing of rules. Second, a pattern $P1_{\mathrm{Y}}$ subsumes a pattern $P2_{\mathrm{X}}$ if and only if Equation 1 holds for **all possible interpretations** of $\mathcal{T}$. For instance, suppose we define a House-owner to be a person who owns at least a house. The pattern (Own-house ?x ?y) does not subsume the pattern (House-owner ?x) because a match with the latter does not guarantee a match with the former due to incompleteness of the knowledge base (e.g., the system may know John owns some houses without knowing any specific houses that are owned by him). Third, the definition allows the parent pattern to contain extra conditions that do not have counterparts in the child pattern. For example, the pattern `(robot ?x) ∨ (animal ?x)` subsumes the pattern `(super-robot ?robot)` under the substitution (?robot/?x) even though the condition about `(animal ?x)` does not have a more specific counterpart in the child pattern.

To determine whether a pattern $P1_{\mathrm{Y}}$ subsumes another pattern $P2_{\mathrm{X}}$, we need to find a substitution that replaces variables in $P1_{\mathrm{Y}}$ by arguments in $P2_{\mathrm{X}}$ such that the latter terminologically implies the former under the substitution. *Terminological implication*, denoted as $\overset{\mathcal{T}}{\Rightarrow}$, is defined as follows: $P2_{\mathrm{X}} \overset{\mathcal{T}}{\Rightarrow} P1_{\mathrm{Y}}$ iff

$$\forall \mathcal{I}_{\mathcal{T}} = (\mathcal{D}, \mathcal{E})$$
$$\left[\forall \vec{x} \in \mathcal{D}^n\ \left(P2_{\mathrm{X}}{}^{\mathcal{E}}\ (\vec{x}) \Rightarrow \{p1_{\mathrm{X'}}{}^{\mathcal{E}}\ (\vec{x'})\right)\right] \qquad (2)$$

where the set of variables in X' is a subset of variables in X, and hence the variable binding of X' is directly obtained from that of X. More formally, we have the following Theorem.

Theorem 1 *Suppose patterns* $P1_{\mathrm{Y}}$ *and* $P2_{\mathrm{X}}$ *are boolean combinations of literals. The pattern* $P1_{\mathrm{Y}}$ *subsumes* $P2_{\mathrm{X}}$ *iff there exists a* subsumption substitution *S that replaces variables of* $P1_{\mathrm{Y}}$ *by* $P2_{\mathrm{X}}$*'s variables or constants such that* $P2_{\mathrm{X}}$ *terminologically implies* $P1_{\mathrm{Y}}/S$ *based on the terminological knowledge base* $\mathcal{T}$, *i.e.,*

$$P1_{\mathrm{Y}} \succeq P2_{\mathrm{X}}\ \textit{iff}\ \exists S\ \textit{such that}\ P2_{\mathrm{X}} \overset{\mathcal{T}}{\Rightarrow} P1_{\mathrm{Y}}/S. \qquad (3)$$

The proof of the theorem is based on skolemizing the existentially quantified variable $\vec{y}$ in Equation 1. Detail of the proof can be found in [Yen 1990].

The subsumption substitution S can also be viewed as a *mapping* because it maps a variable of of pattern $P1_{\mathrm{Y}}$ to a variable or a constant in pattern $P2_{\mathrm{X}}$. We will use the terms "subsumption substitution" and "subsumption mapping" interchangeably in our discussion. Intuitively, it is easy to see that the existence of

[2] When it is not important to refer to the variables of a pattern, we denote patterns simply by P.

P1: `(:and (father ?x ?y) (father ?x ?z) )`
P2: `(father ?u ?v)`

Figure 1: An Example of Two Indifferent Patterns

a subsumption mapping is a sufficient condition that $P2_X$ is more specific than $P1_Y$ because, for any instantiation of $P2_X$'s variables, we can construct an instantiation of $P1_Y$'s variables from the subsumption mapping. Thus, matching $P2_X$ implies matching $P1_Y$ if a subsumption mapping exists.

We further define the following relationships between patterns:

- Two patterns are *indifferent*, denoted by $\sim$, if and only if they subsume each other, i.e.,

$$P_1 \sim P_2 \Leftrightarrow P_1 \succeq P_2 \wedge P_2 \succeq P_1.$$

  Indifferent patterns are merged in the specificity lattice. Conceptually, two patterns are indifferent if, for any states of the fact database, either both patterns match or neither of them matches the fact database.

- Two patterns are *equivalent*, denoted by $\equiv$, if they are indifferent and the subsumption mapping is a one-to-one mapping between variables of the two patterns. Two indifferent patterns may not be equivalent. For instance, the patterns P1 and P2 in Figure 1 are indifferent because P1 subsumes P2 under the substitution (?u/?x, ?v/?y, ?v/?z) and P2 subsumes P1 under the substitution (?x/?u, ?y/?v) or (?x/?u, ?z/?v). But the two patterns are not equivalent because they have different instantiations for a given facts database.

- Two patterns are *equal*, denoted by $=$, if they are equivalent under a subsumption mapping that maps each variable to a variable with the same name.

The subsumption substitution differs from substitution in unification in that it is directional. It substitutes variables/constants of a child pattern for variables of a parent pattern, but not the other way. This distinction is due to the fact that a subsumption test is meant to test implications, which is directional, while unification is meant to test equality, which is bidirectional.

Classifying Conjunctive Patterns

This section describes an algorithm for classifying patterns that are conjunctions of non-negated literals (which we will refer to as *conjunctive patterns*). The algorithm consists of two steps. First, each pattern is normalized by making explicit in the pattern any unstated conditions logically implied by the patterns and the terminological knowledge. Second, the algorithm searches for a subsumption substitutions between pairs of normalized patterns.

A General Strategy

The general strategy of CLASP's pattern classification algorithm is to simplify the subsumption test between pairs of patterns by first normalizing them. This strategy is analogous to completing a concept definition before actually classifying the concept in KL-ONE's classifier[Schmolze and Lipkis 1983]. To formally justify our approach, this section first defines normalized patterns, then describes a theorem about the subsumption test of normalized conjunctive patterns.

A pattern is normalized if it contains no implicit conditions other than those that can be deduced easily from the subsumption lattice of concepts and of roles, which has been precomputed by LOOM's classifier. More formally, we define a normalized pattern as follows:

Definition 2 *A pattern P is said to be* normalized *iff*

$$\forall l, \text{ if } P \overset{\mathcal{I}}{\Rightarrow} l, \text{ then } \exists l' \text{ in } P \text{ such that } l' \overset{\mathcal{I}}{\Rightarrow} l \quad (4)$$

where l and l's are literals with the same number of arguments.

We say a pattern $\overline{P}$ is a *normalized form* of p if and only if $\overline{P}$ is normalized and P equals $\overline{P}$ (i.e., they are equivalent without variable substitution).

The rationale behind normalizing patterns is to simplify the subsumption test. Without the normalization process, the search for a subsumption substitution would have to consider the possibility that a condition in the parent pattern subsumes a conjunctive subpattern of the child pattern. For example, consider rules **R2** and **R3** in Figure 4. The condition `(College-graduate ?y)` in R2 subsumes the subpattern `(Successful-Father ?z)` $\wedge$ `(Child ?z ?w)` of R3's condition under the substitution `?y/?w`. Having deduced the conditions implied by these conjunctive subpatterns during the normalization process, the subsumption test only needs to consider pairs of conditions with the same arity (one from the parent pattern, one from the child pattern) for testing subsumption possibility of the two patterns. Thus, normalizing patterns significantly reduces the complexity of the subsumption test. The following theorem formally states the impact of pattern normalization to the subsumption test.

Theorem 2 *Suppose $P1$ and $P2$ are two normalized conjunctive patterns:*

$$P1 = l_1^1 \wedge l_2^1 \wedge ... l_n^1 \quad (5)$$

$$P2 = l_1^2 \wedge l_2^2 \wedge ... l_m^2 \quad (6)$$

where l_i^1 and l_j^2 are literals without negations. The pattern $P1$ subsumes $P2$ if and only if there exists a subsumption substitution S such that every literal l_i^1 in $P1$ subsumes at least one literal in $P2$ with the same arity, i.e.,

$$P1 \succeq P2 \Leftrightarrow$$

$$\exists S \left[\forall l_i^1 \text{ in } P1, \exists l_j^2 \text{ in } P2, \text{ such that } l_j^2 \overset{\mathcal{I}}{\Rightarrow} l_i^1/S \right] \quad (7)$$

where l_i^1 and l_j^2 have the same number of arguments.

To prove the theorem, we first introduce the following lemma.

Lemma 1 *Suppose $P1$ is a conjunction of n literals, i.e., $P1 = l_1^1 \wedge l_2^1 \wedge ...l_n^1$, where $l_1^1, l_2^1, ...l_n^1$ are literals without negations. The pattern $P1$ subsumes a pattern $P2$ if and only if there exists a subsumption substitution such that each literals l_i^1 subsumes the pattern $P2$ under the substitution, i.e.,*

$$P1 \succeq P2 \Leftrightarrow \exists S \text{ such that}$$

$$(P2 \stackrel{\mathcal{T}}{\Rightarrow} l_1^1/S) \wedge (P2 \stackrel{\mathcal{T}}{\Rightarrow} l_2^1/S) \wedge ...(P2 \stackrel{\mathcal{T}}{\Rightarrow} l_n^1/S) \quad (8)$$

Proof of Lemma 1 and Theorem 2 can be found in [Yen 1990].

Comparing Equations 3 and 7, we can see immediately that the complexity of the subsumption test has been reduced significantly by first normalizing the patterns. Based on Theorem 2, computing whether $P2$ is more specific than $P1$ only requires searching for a *subsumption mapping* such that each condition (i.e., literal) in $P1$ subsumes at least one condition (i.e., literal) in $P2$ under the mapping. We will refer to a $P2$'s condition that is subsumed by a condition l_1^i in $P1$ as l_1^i's *subsumee*. The subsumption test between normalized conjunctive patterns, thus, is a simpler search problem. The following two sections describe the normalization of patterns and the subsumption test between normalized patterns implemented in CLASP.

Normalizing Patterns

The normalization step transforms each pattern into an equivalent normalized pattern. Five kinds of normalization steps have been implemented in CLASP: (1) domain and range deductions, (2) normalizing unary conditions, (3) normalizing binary conditions, (4) value restriction deductions, and (5) at-least-one deductions. Each normalization step will be described and illustrated with examples, based on Figures 3 and 4. These normalization steps are correct because each one transforms a pattern into an equivalent one based on the semantics of LOOM's term-forming expressions in Figure 2.

1. *Domain and Range Deduction:* This step deduces unary conditions about variables that appear in a binary condition using domains and ranges of the condition's predicate (i.e., a relation). For instance, this step will infer an implicit condition of R3 `(Vehicle ?c)` from the range of `Has-car` relation.
2. *Normalizing Unary Conditions:* Unary conditions that involve the same variables are replaced by one unary condition whose predicate is the conjunction of the unary predicates (i.e., concepts) in the original pattern. This ensures that all patterns are transformed into a canonical form where each variable has at most one unary condition. The condition-side of R2 thus is normalized to combine two unary conditions about the variable ?x. into one condition `(College-graduate&Car-Owner ?y)` where `College-graduate&Car-Owner` is the conjunct of `College-graduate` and `Car-Owner`.

```
(defrule R2
        :when (:and (College-graduate ?y)
                    (Child ?x ?y)
                    (Car-Owner ?y))
        ... )

(defrule R3
        :when (:and (Successful-Father ?z)
                    (Father ?f)
                    (Child ?z ?w)
                    (Son ?f Fred)
                    (Female ?w)
                    (Has-Car ?w ?c))
        ... )
```

Figure 4: Two Rules Before Normalization

3. *Normalizing Binary Conditions:* Binary conditions with the same arguments are collected, and replaced by a new composite binary condition that takes into account the unary conditions of its domain variable and its range variable. This ensures that all normalized patterns have at most two binary conditions for each variable pair (the argument position of the variables can be switched). For instance, the conditions in R3 `(Child ?z ?w)` $\wedge$ `(Female ?w)` are transformed to `(Daughter ?z ?w)` $\wedge$ `(Female ?w)`.
4. *Value Restriction Deduction:* Suppose a pattern contains conditions of the form `(:and (`C_1 `?x) (R ?x ?y) ... )` and the definition of C_1 in the terminological space has a value restriction on R, say C_2. Then the pattern is equivalent to a pattern that has an additional unary condition (C_2 ?y). For example, conditions `(Successful-Father ?z)` and `(Daughter ?z ?w)` in R3 deduce an implicit condition `(College-graduate ?w)` because successful-father has been defined as a father all whose children, which include daughters, are college graduates as shown in Figure 3 .
5. *At-least-one Deduction:* A pattern containing two conditions in the form of `(:and ... (C ?x) ... (R ?x` α`) ...)`, where α is either a variable or a constant, is transformed to one that replaces C by the concept C' defined below, which has an additional at-least-one number restriction on the relation R.
`(defconcept C' (:and C (:at-least 1 R)))`
Following our example, the conditions `(Female ?w)` and `(Has-car ?w ?c)` in R3 now can deduce another implicit condition about ?w: `(Car-owner ?w)`, for `Car-Owner` has been defined to be a person who

Expression e	*Interpretation* $[\![e]\!]$
(:and C_1 C_2)	$\lambda x.\, [\![C_1]\!](x) \wedge [\![C_2]\!](x)$
(:and R_1 R_2)	$\lambda xy.\, [\![R_1]\!](x,y) \wedge [\![R_2]\!](x,y)$
(:at-least 1 R)	$\lambda x.\, \exists y.\, [\![R]\!](x,y)$
(:all R C)	$\lambda x.\, \forall y.\, [\![R]\!](x,y) \rightarrow [\![C]\!](y)$
(:domain C)	$\lambda xy.\, [\![C]\!](x)$
(:range C)	$\lambda xy.\, [\![C]\!](y)$

Figure 2: Semantics of Some Term-Forming Expressions

```
(defconcept Person (:primitive))
(defconcept Male (:and Person :primitive))
(defconcept Female (:and Person :primitive))
(defconcept College-graduate (:and Person :primitive))
(defrelation Child (:and :primitive (:domain Person) (:range Person)))
(defrelation Daughter (:and Child (:range Female)))
(defrelation Son (:and Child (:range Male)))
(defconcept Father (:and Male (:at-least 1 Child )))
(defconcept Successful-Father (:and Father (:all Child College-graduate)))
(defrelation Has-car (:and :primitive (:domain Person) (:range Vehicle)))
(defconcept Car-owner (:and Person (:at-least 1 Has-car)))
```

Figure 3: An Example of Terminological Knowledge

```
(defrule R2
   :when
     (:and (Person ?x)
           (College-graduate&Car-Owner ?y)
           (Child ?x ?y))
        ... )

(defrule R3
   :when
     (:and (Successful-Father ?z)
           (Female&College-graduate&Car-owner ?w)
           (Daughter ?z ?w)
           (Father  ?f)
           (Vehicle ?c)
           (Son ?f Fred)
           (Has-Car ?w ?c))
        ... )
```

Figure 5: Two rules after normalization

has at least one car.

Figure 5 shows the condition-sides of R2 and R3 after they have been normalized. It is easier to see that R3 is actually more specific than R2, which was not obvious prior to normalization.

Testing Subsumption of Normalized Conjunctive Patterns

Reducing the Search Space Although an exhaustive search that considers all possible mappings can not be avoided in the worst case, the search space of possible subsumption mapping can be significantly reduced in most cases by considering the subsumption relationship between predicates. Normally, the condition pattern of a rule consists of several different predicates, only a small percentage of which are subsumed by a predicate in another pattern. Thus, using the subsumption relationships between predicates, we can significantly reduce the the search space for finding a subsumption mapping.

Our strategy is to identify *potential subsumees* for all literals in the parent pattern $P1$. A literal l_2 is a *potential subsumee* of a literal l_1 if there exists a subsumption substitution S such that $l_2 \stackrel{\mathcal{T}}{\Rightarrow} l_1/S$. The set of potential subsumees of a unary literal determines a set of potential candidates (which we call *potential images*) that a variable can map to under a subsumption mapping. The set of potential subsumees of a binary literal generates *mapping constraints* on how pairs of variables should be mapped. Potential images are used to reduce the branching factor of the search space, and mapping constraints are used to prune the search tree. This is illustrated using the example in Figure 5. Only two conditions in R3, `(Son ?x Fred)` and `(Daughter ?z ?w)`, can potentially be subsumed by `(Cdhild ?x ?y)` in R2. Since `(Child ?x ?y)` must have a subsumee under a subsumption mapping, we can infer that any subsumption mapping that proves

R3 is more specific than R2 has to satisfy one of the following two mapping constraints: (1) If `(Child ?x ?y)` subsumes `(Son ?f Fred)`, then the variable `?x` should map to `?f` and the variable`?y` should map to `Fred`. (2) If `(Child ?x ?y)` subsumes `(Daughter ?z ?w)`, then the variable `?x` should map to `?z` and the variable `?y` should map to `?w`. Similarly, potential subsumees of a parent pattern's unary condition restrict the candidate images a variable can map to. Using the example in Figure 5 again, `(Successful-father ?z)` and `(Father ?f)` are the only two unary conditions in R3 that can potentially be subsumed by `(Person ?x)` in R2. Hence, the potential images of `?x` are `?z` and `?f`.

The process of reducing the search space can also detect early failure of the subsumption test. The subsumption test terminates and returns false whenever (1) it fails to find any potential images for a variable in $P2$; or (2) a binary condition in $P1$ fails to find any binary condition in $P2$ as a potential subsumee.

Searching for a Subsumption Substitution A subsumption mapping between two normalized patterns is constrained by the potential images of each variables in the parent pattern and the mapping constraints imposed by binary conditions of the parent pattern $P1$. To search for a subsumption mapping that satisfies these constraints, which are generated by algorithms discussed in previous sections, the pattern classifier first sorts the parent variables in increasing order of the number of their potential images, then performs a dependency-directed backtracking. The position of a variable in the sorted list corresponds to the level it's images are assigned in the search tree. At each node, the current assignment of variables' images are checked to see if it satisfies the mapping constraints. If it does not satisfy any of the mapping constraint, the algorithm backtrack to the closest node whose assignment causes a constraint violation.

Discussion

We have shown elsewhere that CLASP's pattern classification algorithm is sound [Yen 1990]. It is also complete for a simple term subsumption language whose expressiveness is equivalent to that of $\mathcal{FL}^-$ in [Brachman and Levesque 1984]. In general, an implementation of our pattern classification algorithm is sound if (1) the normalization step transforms an input pattern to an equivalent one, and (2) all identified potential subsumees are correct (which requires the classifier to be sound). An implementation of the general algorithm is complete if (1) the normalization step transforms an input pattern into its normalized equivalent form, and (2) the complete set of potential subsumees are identified for each literals of the parent pattern (which requires the classifier to be complete). A more detailed discussion on the issues regarding soundness and completeness of the pattern classification algorithm can be found in [Yen 1990].

Determining the subsumption of normalized conjunctive patterns is NP-complete, for it can be reduced from the problem of determining subgraph isomorphism for directed graphs, which is known to be NP-complete. However, worst case rarely occur in practice. To analyze the behavior of an algorithm in reality, we have defined *normal cases*[3] and have shown that the complexity of the algorithm for normal cases is polynomial [Yen 1990].

Brachman and Levesque have demonstrated that there is an important tradeoff between the expressiveness of a terminological language and the complexity of its reasoner [Brachman and Levesque 1984]. A similarly tradeoff between the computational complexity of the normalization process and the expressiveness of the terminological language has also been investigated [Yen 1990].

Summary

We have presented a principled approach to computing the specificity of rules whose conditions are constructed from terms defined using a terminological language. Based on a formal definition of pattern subsumption relation, we first show that the subsumption test between conjunctive patterns can be viewed as a search problem. Then we describe a pattern classification algorithm that improves the efficiency of the search process in two ways. First, implicit conditions logically implied by a pattern is made explicit before the search step. Second, the algorithm attempts to reduce the search space using information about the subsumption relation between predicates.

Our approach offers several important benefits to the developers of rule-based systems. First, the pattern classifier makes it possible to provide, for the first time, a principled account of the notion of rule-specificity as a guide to conflict resolution. This will greatly improves the predictability of rule-based systems, and thus alleviate the problems in maintaining them. Second, using pattern classifier to compute the specificity of methods, CLASP is able to generalize methods in object-oriented programming for describing a complex situation to which the method applies. Third, separating definitional knowledge from rules enhances the reusability of knowledge and the explanation capability of the system. Finally, the pattern classifier is also the enabling technology for our future development of a rule base organizer, which automatically determines groupings of a large set of rules based on the semantics of rules and rule classes.

Acknowledgements

I would like to thank Robert Neches for his encouragement and support of this research. I am also grate-

[3] Using normal cases to analyze the complexity of intractable algorithm has been suggested by Bernard Nebel [Nebel 1989].

ful to Robert MacGregor, Bill Swartout, and David Benjamin for their fruitful ideas regarding the the pattern classification algorithm. Finally, the research on CLASP has benefited from many discussions with Paul Rosenbloom and John Granacki. Part of the work described in this paper was supported by Engineering Excellence Fund at Texas A&M University.

References

[Brachman and Levesque, 1984] Ronald J. Brachman and Hector J. Levesque. The tractability of subsumption in frame-based description languages. In *Proceedings of AAAI-84*, pages 34–37, Austin, Texas, August 1984.

[Brachman and Schmolze, 1985] R.J. Brachman and J.G. Schmolze. An overview of the kl-one knowledge representation system. *Cognitive Science*, pages 171–216, August 1985.

[Fikes and Kehler, 1985] R. Fikes and T. Kehler. The role of frame-based representation in reasoning. *Communication of the ACM*, 28(9), September 1985.

[Gregor and Bates, 1987] Robert Mac Gregor and Raymond Bates. The loom knowledge representation language. Technical Report ISI/RS-87-188, USC/Information Sciences Institute, 1987.

[Mark, 1981] William Mark. Representation and inference in the consul system. In *Proceedings of the Seventh International Joint Conference on Artificial Intelligence*, pages 375–381. IJCAI, Morgan Kaufman, 1981.

[McDermott and Forgy, 1978] J. McDermott and C. Forgy. Production system conflict resolution strategies. In D. A. Waterman and F. Hayes-Roth, editors, *Pattern-Directed Inference Systems*. Academic Press, New York, 1978.

[Nebel, 1989] Bernhard Nebel. Terminological reasoning is inherently intractable. Technical Report IWBS Report 82, IWBS, IBM Deutschland, W. Germany, October 1989.

[Nguyen *et al.*, 1985] T. A. Nguyen, W. A. Perkins, and T. J. Laffey. Checking an expert system knowledge base for consistency and completeness. In *Proceedings of IJCAI-85*, pages 375–378, Los Angeles, CA, August 1985.

[Patel-Schneider *et al.*, 1990] Peter F. Patel-Schneider, Bernd Owsnicki-Klewe, Alfred Kobsa, Nicola Guarino, Robert MacGregor, William S. Mark, Deborah McGuinness, Bernhard Nebel, Albrecht Schmiedel, and John Yen. Report on the workshop on term subsumption languages in knowledge representation. to appear in AI Magazine, 1990.

[Schmolze and Lipkis, 1983] James Schmolze and Thomas Lipkis. Classification in the kl-one knowledge representation system. In *Proceedings of the Eighth International Joint Conference on Artificial Intelligence*. IJCAI, 1983.

[Suwa *et al.*, 1982] Motoi Suwa, A. Carlisle Scott, and Edward H. Shortliffe. An approach to verifying completeness and consistency in a rule-based expert system. *AI Magazine*, 3(4):16–21, Fall 1982.

[Swartout and Neches, 1986] William Swartout and Robert Neches. The shifting terminological space: An impediment to evolvability. In *AAAI-86, Proceedings of the National Conference on Artificial Intelligence*, Philadelphia, PA, August 1986. AAAI.

[Wellman, 1988] Michael P. Wellman. *Formulation of Tradeoffs in Planning Under Uncertainty*. PhD thesis, Massachusetts Institute of Technology, Department of Electrical Engineering and Computer Science, 1988. Also distributed as a Computer Science Laboratory technical report MIT/LCS/TR-427.

[Yen *et al.*, 1989] John Yen, Robert Neches, and Robert MacGregor. Using terminological models to enhance the rule-based paradigm. In *Proceedings of the Second International Symposium on Artificial Intelligence*, Monterrey, Mexico, October 25-27 1989.

[Yen, 1990] John Yen. Reasoning about specificity of patterns in term subsumption-based systems. Technical Report TAMU 90-003, Department of Computer Science, Texas A&M University, February 1990.